15-451/651 Algorithms, Fall 2017

Homework #4 **Due: October 18-22, 2017**

This is an oral presentation assignment. Again, there are three regular problems (#1-#3) and one programming problem (#4). You should work in groups of three. The presentation sign-up sheet will be online soon (details on Piazza), and your group should sign up for a 45 minute slot. Each person in the group must be able to present every regular problem. The TA/Professor will select who presents which regular problem. You are not required to hand anything in at your presentation, but you may if you choose.

The programming problem will be submitted to autolab, similar to HW#1. It is due at 11:59pm on Tuesday, October 24. You will not have to present anything orally for the programming problem. Please include a comment in your program explaining the algorithm you used. You can discuss the problem with your group-mates, but must write the program by yourself. Please do not copy.

Problem #B2 is a "bonus" problem. It is optional, and you won't get hints or assistance from the course staff. It is worth **1%** under the reci/quiz/bonus category if you solve it correctly. No partial credit, sorry. The solution to this problem must be written up and submitted via `gradescope` by October 24.

---

1. **Counts of Substrings**

   Assume all strings are over a constant-sized alphabet and are indexed starting at 0.

   (a) Let x be a string of length n. There are $O(n^2)$ substrings of x, but some of them may have the same sequence of characters, of course. Show how to count the number of *distinct* substrings of x in $O(n)$ time.

   (b) In a string S, a *maximal repeat occurrence* is a pair of positions i and j in S and a length k such that:

   i. $S[i, \ldots, i+k-1] = S[j, \ldots, j+k-1]$,
   ii. $S[i-1] \neq S[j-1]$,
   iii. $S[i+k] \neq S[j+k]$

   Above, we take $S[-1]$ and $S[n]$ to be symbols distinct from every other symbol. A *maximal repeat string* is a string w such that there is a maximal repeat occurrence (i, j) with $S[i, \ldots, i+|w|-1] = w$.

   (i) Give an example of a string in which two different maximal repeat strings both start at the same position. In other words, find a string with two maximal repeat occurances (i, j, k) and (i, j', k').

   (ii) Using suffix trees, give a short, elegant proof that there are $\leq n$ distinct maximal repeat strings.

2. **Farthest Apart**

You're given an unrooted tree of n nodes. Some of the nodes are marked. Let `mark(x)` be a boolean valued function which is true if node x is marked and false otherwise. At least two nodes are marked.

(a) Give an algorithm that runs in time $O(n)$ to compute the maximum distance between a pair of marked nodes. The distance is just the number of edges in the tree on the path between the two nodes.

(b) Explain how to modify the algorithm in part (a) to find the pair that are farthest apart.

3. **Cycles, articulation points, and separating sets**

(a) Let G be an undirected graph with no self-loops or multi-edges. Prove that if every vertex has degree at least $d \geq 2$, then G contains a simple cycle[1] of length at least $d + 1$. (Hint: Think about what happens if you run the DFS algorithm from lecture on this graph.)

(b) Let $G = (V, E)$ be a connected, undirected graph, and suppose $T_s = (V, F)$ is the tree of "tree edges" obtained when doing a DFS on G starting from vertex s. Vertex s is, of course, the root of tree T_s. Argue that s has more than one child in T_s if and only if removing s from G breaks G into several disconnected parts.

[1] A cycle is *simple* if it never visits the same node twice.

4. **Rep-Count Compression**

"Rep-Count" notation is a way to represent a string of lower case letters (`a` through `z`) with (possibly) fewer characters. A number n in the notation is used to indicate that the following character (or multi-character string surrounded by parentheses) is to be repeated n times.

For example the string `5a` represents `aaaaa`. And `3(hello)` represents `hellohellohello`. Of course this notation can be applied in nested fashion so `3(x3a)` represents `xaaaxaaaxaaa`.

The task is to write a program to take as input a string S of lower-case letters of length at most 500, and compute a shortest Rep-Count representation of S. The output will consist of two lines. The first is the length of the shortest representaton of S, and the second is a representation that is of that length. The representations will only account for 4 points out of 25 points for the problem. If all are correct you will get the 4 points, otherwise you will get 0/4 for that part. The judge log will inform you for each of the inputs if the representation is correct or not. The time limit is 5 seconds.

In the following examples, the input is on the left and the output is on the right.

`hellloooooo`	`6` `he3l6o`
`couscousiscouscousiscouscous`	`19` `2(cous)2(is2(cous))`
`bookkeeper`	`10` `bookkeeper`
`tctttttttttttctttttttttttctttttttttttct`	`14` `tc11tc2(t9tct)`

B2 **(OPTIONAL BONUS PROBLEM)** A *top-dog repeat string* is a maximal repeat string (as in problem 2) that is not a substring of any other maximal repeat string. Example:

```
abcwqabcxabcyqabcz
0123456789012345 67
0          1
```

Above, $(i = 0, j = 5, k = 3)$ is a maximal repeat occurrence and $w =$ `abc` is a maximal repeat string, but it is not a top-dog repeat string since it is a substring of `qabc` (which is a top-dog repeat).

Let S be a string of length n over a constant-sized alphabet. Suppose you have only enough memory to hold S, the suffix array of S, the LCP array of the suffix array, and a constant (independent of n) number of other variables.

Give a $O(n + m)$-time algorithm to print out all the top-dog repeat strings, where m is the total length of the output strings. Again, assume the alphabet size is constant.